Cybernetics and Systems Analysis, Vol. 44, No. 2, 2008

METHODS OF CONSTRUCTING BAYESIAN NETWORKS BASED ON SCORING FUNCTIONS

M. Z. Zgurovskii,[a] P. I. Bidyuk,[b] and A. N. Terent'ev[c]

UDC 62-50

Available methods of constructing Bayesian networks with the use of scoring functions are analyzed. The Cooper–Herskovits and MDL functions are described in detail and used to compare algorithms of constructing Bayesian networks.

Keywords: *Bayesian network, search and scoring method, computational characteristics, minimum description length.*

INTRODUCTION

A great amount of information that require adequate processing and decision-making based on the results of this processing is accumulated worldwide. Methods of intelligent data analysis (IDA), which include Bayesian networks (BNs), afford an opportunity of automatic search for the principles characteristic of multidimensional data. The majority of tools for intelligent data analysis underlie two techniques: machine learning and data visualization. Bayesian networks combine both of them.

Bayesian networks are widely used in medical and engineering diagnostics under incomplete and imperfect information, in data classification systems, automatic speech recognition, marketing, business, and many other activities. In the general case, BNs make it possible to reproduce cause-effect relationships among events and to determine event probability if new information on changes in the state of any node (variable) of the network becomes available. The degree of expediency of applying this method of modeling and probabilistic inference depends on the ability to formulate the problem correctly, to select variables of the process that characterize its dynamics or statics in a sufficient degree, to find necessary data and use them for network learning, and formulate correctly the resulting inference using the network constructed.

In the English-speaking literature, the term "construction of a BN" means implementing the following processes: (i) searching for the optimal structure, i.e., a directed acyclic graph that most adequately fits the learning data or the process under study; (ii) computing the values of conditional probability tables of the BN for the corresponding nodes of this graph.

The purpose of this paper is to analyze the methods available to solve the problem of choosing the optimal structure of a Bayesian network and to describe their intrinsic principles and practical application.

FORMAL MATHEMATICAL NOTATION OF A BN

A Bayesian network is a graphical model of a process or an object of arbitrary nature, represented by a pair $< G, B >$. The first component G is a directed acyclic graph that corresponds to the random variables of the object or process. It is written as a set of independence conditions: each variable does not depend on its parents in G. The second component B is a set of parameters that define the network. This component contains parameters $\Theta_{X^{(i)}|pa(X^i)} = P(X^{(i)} | pa(X^{(i)}))$ for each

[a]National Technical University "Kyiv Polytechnical Institute," Kyiv, Ukraine; Institute of Applied Systems Analysis of the National Academy of Sciences of Ukraine and Ministry of Education and Science of Ukraine, Kyiv, Ukraine, *zgur@zgurov.kiev.ua* and *zgurovsm@hotmail.com*. [b]National Technical University "Kyiv Polytechnical Institute," Kyiv, Ukraine, *peterb@mmsa.ntu-kpi.kiev.ua.* [c]Institute of Applied Systems Analysis of the National Academy of Sciences of Ukraine and Ministry of Education and Science of Ukraine, Kyiv, Ukraine, *kenga@voliacable.com*. Translated from Kibernetika i Sistemnyi Analiz, No. 2, pp. 81–88, March–April 2008. Original article submitted June 27, 2007.

1060-0396/08/4402-0219

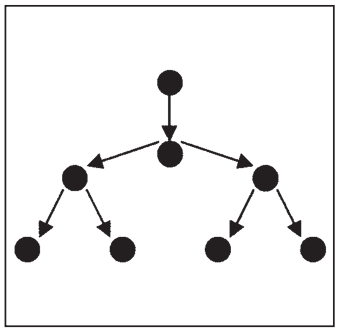

Fig. 1. A tree BN structure.

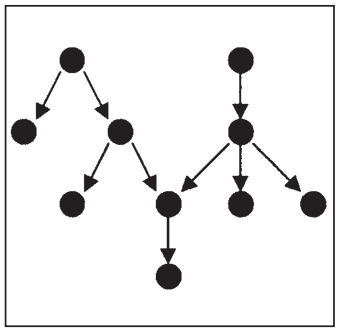

Fig. 2. A polytree BN structure.

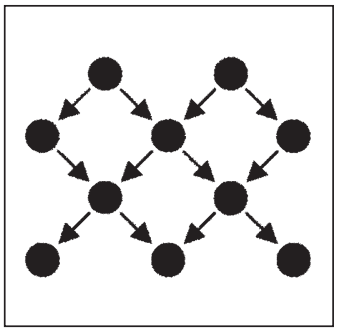

Fig. 3. A network BN structure.

possible value $x^{(i)} \in X^{(i)}$ and $pa(X^{(i)}) \in Pa(X^{(i)})$, where $Pa(X^{(i)})$ means the set of parents of the variable $X^{(i)} \in G$. Each variable $X^{(i)} \in G$ is represented as a node. If more than one graph is considered, the notation $Pa^G(X^{(i)})$ is used to define parents of the variable $X^{(i)}$ in the graph *G*. The total joint probability of the BN can be calculated by the formula $P_B(X^{(1)}, \ldots, X^{(N)}) = \prod_{i=1}^{N} P_B(X^{(i)} | Pa(X^{(i)}))$.

The set of learning data can be written as follows: $D = \{d_1, \ldots, d_n\}$, $d_i = \{x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(N)}\}$, where the subscript is the number of observation, and the superscript is the number of the variable; *n* is the number of observations, each consisting of *N* ($N \geq 2$) variables $X^{(1)}, X^{(2)}, \ldots, X^{(N)}$. Each *j*th variable ($j = 1, \ldots, N$) has $A^{(j)} = \{0, 1, \ldots, \alpha^{(j)} - 1\}$ ($\alpha^{(j)} \geq 2$) states, and each structure $g \in G$ of the BN can be represented by *N* sets of parents ($\Pi^{(1)}, \ldots, \Pi^{(N)}$). In other words, for each node $j = 1, \ldots, N$, the set $\Pi^{(j)}$ is a set of parent nodes such that $\Pi^{(j)} \subseteq \{X^{(1)}, \ldots, X^{(N)}\} \setminus \{X^j\}$ (a node cannot be a parent for itself, i.e., there are no closed loops in the graph).

GRAPHICAL REPRESENTATION OF BN STRUCTURE

A tree is a BN structure such that any node can have no more than one parent node (Fig. 1).

A polytree is a BN structure such that any node can have more than one parent node but any two nodes should have no more than one path connecting them (Fig. 2).

Networks are a network structure where any node can have more than one parent node and any two nodes can have more than one path connecting them (Fig. 3).

Trees and polytrees are also called simply connected networks, and networks are called multiply connected networks.

ALGORITHMS OF CONSTRUCTING BN STRUCTURE

In 1968, Chow and Liu proposed an algorithm to construct a BN as a tree [1]. The algorithm is based on using the values of mutual information between nodes. As a solution, the method produces the BN structure with the value of the joint probability distribution that best fits the learning data. The BN structure is constructed in $O(N^2)$ steps, where *N* is the number of network nodes; however, this algorithm is inapplicable for multiply connected BNs.

In 1988, Rebane and Pearl proposed an advanced modified Chow–Liu algorithm to construct a BN as a polytree [2]. In 1990, Cooper and Herskovits developed the Kutato algorithm [3]. At the initialization stage, assuming that all nodes of the BN are independent, the entropy of this network is calculated. Then arcs between nodes are added in the network so as to minimize the BN entropy. This algorithm requires node ordering.

The SGS algorithm [4] proposed in 1991 does without an ordered set of nodes; however, it should instead execute an exponential number of tests for conditional independence between nodes. In 1992, Cooper and Herskovits proposed the well-known K2 algorithm [5], which searched for a structure with the maximum value of the Cooper–Herskovits (CH) function. This algorithm needs an ordered set of nodes.

The Lam–Bacchus algorithm [6], proposed in 1996, fulfills the heuristic construction of the network structure using the mutual information between nodes, and the minimum description length (MDL) as a scoring function.

The Benedict algorithm [7], proposed in 1996, employs the heuristic search based on an ordered set of nodes, analyzes conditional independences in the network structure based on *d*-separation, and uses entropy as a scoring function.

The CB algorithm [8] was proposed in 1995. It uses a test for conditional independence between network nodes to construct an ordered set of nodes. The CH function is used to construct the network structure.

The Friedman–Goldszmidt algorithm [9] was proposed in 1996. To construct a network, its local substructures are analyzed, and MDL and Bayesian estimate are used as scoring functions.

In the WKD algorithm [10], proposed in 1996, the function of minimum message length, similar to MDL, is used as a scoring function to construct a network.

The Suzuki algorithm [11], proposed in 1999, is based on the branch and bound algorithm to represent the sequence of constructing the network structure, and MDL is used as a scoring function.

COOPER–HERSKOVITS FUNCTION

In [5], Cooper and Herskovits proposed the CH method for BN learning based on searching for a BN structure with the maximum value of the CH function. For a given sequence of *n*-observations $x^n = d_1 d_2 \dots d_n$, the CH function of structure $g \in G$ can be written by the equation

$$P(g, x^n) = P(g) \cdot \prod_{j \in J} \left(\prod_{s \in S(j,g)} \frac{(\alpha^{(j)} - 1)! \cdot \prod_{q \in A^{(j)}} (n[q,s,j,g]!)}{(n[s,j,g] + \alpha^{(j)} - 1)!} \right),$$

where $P(g)$ is a priori probability of the structure $g \in G$, which is often omitted in computations; the notation $j \in J = \{1, \dots, N\}$ means the enumeration of all nodes of structure of the network g, and $s \in S(j,g)$ is the enumeration of the set of all sets of values taken by parents of the jth node;

$$n(s,j,g) = \sum_{i=1}^{n} I(\pi_i^{(j)} = s); \; n[q,s,j,g] = \sum_{i=1}^{n} I(x_i = q, \pi_i^{(j)} = s),$$

where $\pi^{(j)} = \Pi^{(j)}$, the function $I(E) = 1$ when the predicate $E =$ true, otherwise $I(E) = 0$.

The BN learning algorithm using the CH function is based on cyclic enumeration of all possible acyclic network structures. The optimal network structure remains in g^*. The optimal structure is the one with the greatest value of the function $P(g, x^n)$:

(i) $g^* \leftarrow g_0 (\in G)$;

(ii) $\forall\, g \in G - \{g_0\}$: if $P(g, x^n) > P(g^*, x^n)$, then $g^* \leftarrow g$;

(iii) at the output, we have g^* as a solution.

However, when using the CH function, it is necessary to account for computational constraints of the analogs due to a finite word length. Let us present a trivial example, where there are two nodes in the structure, $X^{(1)}$ and $X^{(2)}$, and the set of learning examples includes million entries $D = \{d^{(1)}, \dots, d^{(1.000.000)}\}$. Then to determine $P(g, x^n)$, it is required to calculate a factorial $(n[s,j,g] + \alpha^{(j)} - 1)! = (1.000.000 + \alpha^{(j)} - 1)!$ while 32-bit software such as MatLab and MathCAD can calculate factorials no greater than 170!.

MODIFIED LOGARITHMIC COOPER–HERSKOVITS FUNCTION

For a wider use of the CH function, we should get rid of the factorial. To this end, we take the logarithm of the equation that describes the CH function:

$$\log(P(g, x^n)) = \log\left(P(g) \cdot \prod_{j \in J} \left(\prod_{s \in S(j,g)} \frac{(\alpha^{(j)} - 1)! \prod_{q \in A^{(j)}} (n[q,s,j,g]!)}{(n[s,j,g] + \alpha^{(j)} - 1)!} \right) \right)$$

$$= \log (P(g)) + \sum_{j \in J} \left(\sum_{s \in S(j,g)} \left(\sum_{i=1}^{\alpha^{(j)}-1} i + \sum_{q \in A^{(j)}} \left(\sum_{i=1}^{n[q,s,j,g]} i \right) - \sum_{i=1}^{n[s,j,g]+\alpha^{(j)}-1} i \right) \right)$$

$$= \log (P(g)) + \sum_{j \in J} \left(\sum_{s \in S(j,g)} \left(\sum_{q \in A^{(j)}} \left(\sum_{i=1}^{n[q,s,j,g]} i \right) - \sum_{i=\alpha^{(j)}}^{n[s,j,g]+\alpha^{(j)}-1} i \right) \right).$$

Then we multiply the resultant expression by – 1 and, to save computational resources, eliminate $\log (P(g))$. As in [5], we assume that a priori probabilities $P(g)$ of all structures are equal. Instead of searching for a structure with the maximum value of the CH function, we should now search for a structure with the minimum value of the modified logarithmic Cooper–Herskovits function (MLCH):

$$F(g, x^n) = \sum_{j \in J} \left(\sum_{s \in S(j,g)} \left(\sum_{i=\alpha^{(j)}}^{n[s,j,g]+\alpha^{(j)}-1} i \right) \right) - \sum_{j \in J} \left(\sum_{s \in S(j,g)} \left(\sum_{q \in A^{(j)}} \left(\sum_{i=1}^{n[q,s,j,g]} i \right) \right) \right).$$

As computational experiments show, the CH and MLCH functions produce identical solutions on the same learning data. However, on small networks (up to 10 nodes), the algorithm using the CH function operates faster than that using the MLCH, and the situation is opposite on networks with a great number of nodes.

MDL FUNCTION

When constructing a BN, the MDL function [6, 9, 11, 12] or its modifications are often used as a scoring function. For a given sequence $x^n = d_1, d_2, \ldots, d_n$ of n observations, the MDL of a structure $g \in G$ can be calculated by

$$L(g, x^n) = H(g, x^n) + \frac{k(g)}{2} \cdot \log (n),$$

where $k(g)$ is the number of independent conditional probabilities in the network structure g and $H(g, x^n)$ is empirical entropy:

$$H(g, x^n) = \sum_{j \in J} H(j, g, x^n), \; k(g) = \sum_{j \in J} k(j, g).$$

The MDL of the jth node is calculated by the formula

$$L(j, g, x^n) = H(j, g, x^n) + \frac{k(j, g)}{2} \cdot \log (n),$$

where $k(j, g)$ is the number of independent conditional probabilities of the jth node:

$$k(j, g) = (a^{(j)} - 1) \cdot \prod_{k \in \varphi(j)} \alpha^k,$$

$\varphi(j) \subseteq \{1, \ldots, j-1, j+1, \ldots, N\}$ is a set such that $\Pi^{(j)} = \{X^{(k)} : k \in \varphi^{(j)}\}$.

The empirical entropy of the jth node can be calculated from

$$H(j, g, x^n) = \sum_{s \in S(j,g)} \sum_{q \in A^{(j)}} -n[q, s, j, g] \cdot \log \frac{n[q, s, j, g]}{n[s, j, g]};$$

$$n(s, j, g) = \sum_{i=1}^{n} I(\pi_i^{(j)} = s); \; n[q, s, j, g] = \sum_{i=1}^{n} I(x_i = q, \; \pi_i^{(j)} = s),$$

where $\pi^{(j)} = \Pi^{(j)}$ means that $X^{(k)} = x^{(k)} \; \forall \; k \in \varphi^{(j)}$; the function $I(E) = 1$ when the predicate $E = \text{true}$, otherwise $I(E) = 0$.

HEURISTIC SEARCH USING ORDERED SET OF NODES

To reduce the space of network structures, Cooper and Herskovits [5], Dechter [13], and many other contributors [8] propose to assume the set of nodes ordered. In other words, there is an ordered set of nodes $\{X^{(1)}, X^{(2)}, \ldots, X^{(N)}\}$, where $X^{(1)}$ is the principal root node having no parents; $X^{(2)}$ is a child node related to $X^{(1)}$; $X^{(3)}$ is a child node related to any previous node or to all previous nodes simultaneously, etc.

In [5], Cooper and Herskovits propose a heuristic method known in the literature as the K2 algorithm. Parents from $X^{(1)}$ to $X^{(N-1)}$ are added by turn to the node $X^{(N)}$. The CH function is used to calculate $P(g, x^{(n)})$ for each combination. The node $X^{(i)}$ for which $P(g, x^{(n)})$ takes on the maximum value remains to be a parent for the node $X^{(N)}$. After that, parents from $X^{(1)}$ to $X^{(N-2)}$ are added by turn to the nodes $X^{(N)}$ and $X^{(N-1)}$, and $P(g, x^{(n)})$ are calculated. The method outputs the network structure g that maximizes $P(g, x^{(n)})$.

The presence of an ordered set of nodes substantially reduces the space of all possible acyclic structures. However, a new nontrivial problem arises — how to obtain an ordered set of network nodes from a set of learning data. The most obvious way is to involve experts. However, there may be a need to model data in a data domain where there are no qualified experts.

THE VALUE OF MUTUAL INFORMATION BETWEEN NODES

In 1968, Chow and Liu proposed in [1] to use mutual information $MI(x^i, x^j)$ to evaluate the degree of dependence of two arbitrary variables x^i and x^j. The following expression is proposed for calculations:

$$MI(x^i, x^j) = \sum_{x^i, x^j} P(x^i, x^j) \cdot \log\left(\frac{P(x^i, x^j)}{P(x^i) \cdot P(x^j)}\right).$$

By its nature, the value of mutual information is an analog of correlation; by its content, however, it is the estimate of the amount of information on the variable x^j contained in the variable x^i. The value of mutual information is non-negative, $MI(x^i, x^j) \geq 0$, and if the nodes x^i and x^j are completely independent one of the other, $MI(x^i, x^j) = 0$ since $P(x^i, x^j) = P(x^i) \cdot P(x^j)$; therefore,

$$\log\left(\frac{P(x^i, x^j)}{P(x^i) \cdot P(x^j)}\right) = \log\left(\frac{P(x^i) \cdot P(x^j)}{P(x^i) \cdot P(x^j)}\right) = \log(1) = 0.$$

The value of mutual information is used instead of an ordered set of nodes in constructing BNs [1, 2, 6, 12].

RESULTS OF COMPUTATIONAL EXPERIMENTS

The CH and MDL functions and their modifications are most popular scoring functions in constructing BN structures. Therefore, we compare the algorithms that employ MLCH and MDL with respect to the time it takes to construct a BN. To determine the order of adding arcs in a BN, we use the value of mutual information [12], and to construct, we use a genetic data sample consisting of 600 learning entries. Figures 4 and 5 show time expenditures for computations. As is seen, for a BN that consists of more than 30 nodes, the algorithm with the use of MDL operates faster than that with MLCH.

CONCLUSIONS

BNs are widely used in the field of data processing to model processes of various nature and complexity. In the paper, we have analyzed ten methods of constructing BNs with the use of scoring functions. The CH and MDL functions and their modifications are the most popular scoring functions. The results of computational experiments have shown that, on short learning samplings (up to 170 entries) and in the networks consisting of no more than 10 nodes, algorithms that use the CH function operate faster compared with those using the MLCH and MDL. However, algorithms using the MLCH and MDL, in

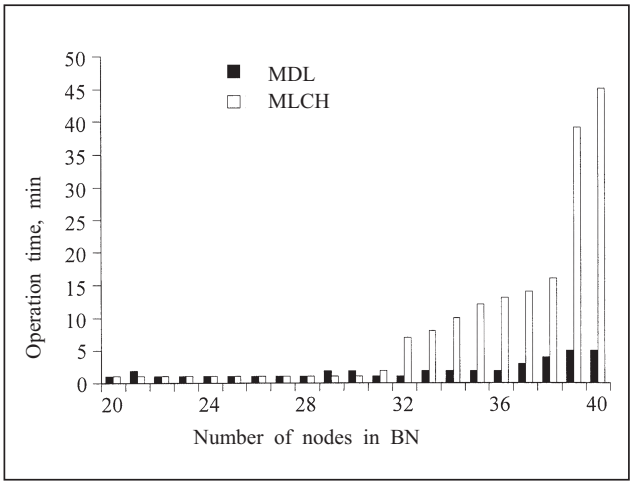

Fig. 4. Computation time for algorithms using MLCH and MDL.

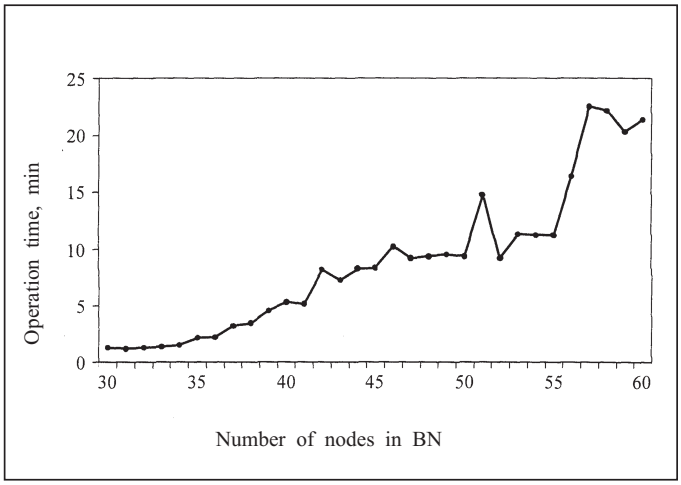

Fig. 5. Time expenditures for algorithm using MDL.

contrast to the CH, operate with learning samples of any size. Computational experiments have shown that methods of constructing BNs with the application of the CH function and its modifications overlearn BNs, i.e., such networks contain redundant arcs.

Compared with the MDL function on genetic data samples consisting of 600 learning entries, the MLCH function shows the best computation time for networks consisting of no more than 30 nodes. But the MDL-based algorithm operates several-fold faster on large networks consisting of more than 30 nodes compared with that using the MLCH function.

REFERENCES

1. C. K. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," IEEE Trans. Inform. Theory, **4**, No. 3, 462–467 (1968).
2. G. Rebane and J. Pearl, "The recovery of causal poly-trees from statistical data," Intern. J. Approx. Res., **2**, No. 3, 175–182 (1988).
3. E. Herskovits and G. Cooper, "Kutato: an entropy-driven system for construction of probabilistic expert systems from databases," in: Proc. 6th Intern. Conf. on Uncertainty in Artificial Intelligence (UAI'90), Cambridge, MA, USA, Elsevier Science, New York (1991), pp. 54–62.
4. P. Spirtes, C. Glymour, and R. Scheines, "From probability to causality," in: Philos. Studies, **64**, No. 1, Springer Netherlands, Amsterdam (1991), pp. 1–36.
5. G. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," Machine Learning, **9**, 309–347 (1992).
6. W. Lam and F. Bacchus, "Learning Bayesian belief networks: an approach based on the MDL principle," Computational Intelligence, **10**, No. 4, 269–293 (1994).
7. S. Acid and L. Campos, "Benedict: an algorithm for learning probabilistic belief networks," in: Proc. 6th Intern. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96), Granada, Spain, Springer, New York (1997), pp. 979–984.
8. M. Singh and M. Valtorta, "Construction of Bayesian network structures from data: a brief survey and an efficient algorithm," Intern. J. Approx. Res., **12**, 111–131 (1995).
9. N. Friedman and M. Goldszmidt, "Learning Bayesian networks with local structure," in: Proc. 12th Intern. Conf. on Uncertainty in Artificial Intelligence (UAI'96), Portland, Oregon, USA, Morgan Kaufmann, SF (1996), pp. 252–262.
10. C. Wallace, K. Korb, and H. Dai, "Causal discovery via MML," in: Proc. 13th Intern. Conf. on Machine Learning (ICML'96), Bari, Italy, Morgan Kaufmann, SF (1996), pp. 516–524.
11. J. Suzuki, "Learning Bayesian belief networks based on the MDL principle: an efficient algorithm using the branch and bound technique," in: IEICE Trans. Inform. Syst. (1999), pp. 356–367.
12. A. N. Terent'ev and P. I. Bidyuk, "A heuristic method to construct Bayesian networks," Mat. Mash. Sist., **3**, 12–23 (2006).
13. R. Dechter, "Bucket elimination: a unifying framework for reasoning," ACM Press, **28,** No. 61, 1–51 (1996).